A Proposal to Extend the Gerber Format: Adding Attributes

Karel Tavernier

August 18th, 2013

Introduction

The Gerber Format is the de facto standard for CAD to CAM data transfer in the PCB industry. It describes the layer images with high precision. Implementations are thoroughly field tested which makes Gerber image transfer very reliable. Untold PCB's - from the simplest ones to the most complex HDI boards - were manufactured using Gerber files.

Note that at the level of the PCB product description drill/route data actually is also image description as it describes where there is material and where not. Indeed, Gerber is used successfully to transfer drill/rout data and manufacturers prefer it over the drill/route formats.

However, to manufacture a bare board PCB more information is needed than just a pile of unidentified images, for example:

1. The function of each image layer (Solder mask top/bottom, copper layer #, legend…)
2. The function of all pads (SMD, via, component,..)
3. ….

This extra information is now transferred informally, with drawing or text files, and not in a formal, standardized machine readable form. The reason for this sad state of affairs is that there is no standard to transfer this information in the context of the ubiquitous Gerber archive.

We propose to extend the Gerber Format with *attributes* to store the extra information associated with the image. This will allow CAD systems not only to transfer the images to CAM system in a standard and machine readable way but also this necessary extra information.

Note that all CAM systems are image based - after all, the image is the heart of a PCB description and by far its most complex component. All CAM systems, however, need also to handle this extra information. CAM systems typically solve this problem by adding *attributes* to the image. Attributes come naturally to a CAM system. It will be a straightforward task for CAM system vendors to add support for Gerber attributes to their systems.

Overview

Attributes do not affect the image, they contain extra information only. Consequently, a Gerber reader that does not process attributes will still generate the correct image. This is very important.

This format extension is upward compatible with the format currently in wide use. ‘Old’ Gerber readers will generate the correct image on new Gerber files.

Attributes can be associated with either

1. The whole file; it then applies to the whole file
2. An aperture; it then applies to all graphic objects created with that aperture
3. A single graphics object; it then applies to that single object

Attributes are defined with the new parameters %TF, %TA and %TO, resp. for file, aperture and object attributes. Details of the syntax and how attributes are associated with objects will be worked out later.

Attributes have a *name* and can take a *value* out of defined range.

Attributes can have the following scope:

1. Standard
2. Custom

Standard attributes are defined in the specification and are part of the Gerber File Format. They extend the format with elements of universal use, such as the pad types or file function. These attribute names begin with a point (“.”), as e.g. in “.FileFunction”.

Custom attributes provide a standard syntax to define custom extensions, e.g. to drive a particular machine. They can and must be ignored by readers of the file not related to that machine. Custom attribute names cannot begin with a period “.”. The custom attributes and their value range must be defined in the header of the Gerber file. The chances for name clash are negligible with well-chosen custom attribute names. A repository of custom name ranges may be organized if the need arises. The meaning or semantics of these attributes is not defined in the Gerber specification but are custom specific.

The File Function Attribute

The **.FileFunction** file attribute identifies the function of the file. It can take the values below.

The copper layers

.FileFunction,Copper,Top,1	Top copper layer.
.FileFunction,Copper,Inn,i	Inner copper layer with index i, where 1<i<n.
.FileFunction,Copper,Bot,n	Bottom copper layer.

The notation is intentionally redundant. Redundancy avoids confusion and makes the system robust.

The attached layers

The attached layers are those functions that are attached to either the top or the bottom. The syntax reflects this common characteristic. For some functions, e.g. solder mask, there can be more than one instance on the same side; this can be indicated with an optional index i>=1.

.FileFunction,Soldermask,{Top|Bot},Index=1?	The solder mask *openings*
.FileFunction,Viafill,{Top|Bot}	
.FileFunction,Legend,{Top|Bot},Index=1?	
.FileFunction,Carbon, {Top|Bot}	
.FileFunction,Goldmask,{Top|Bot}	
.FileFunction,Silvermask,{Top|Bot}	
.FileFunction,Tinmask,{Top|Bot}	
.FileFunction,Peel-off,{Top|Bot}	
.FileFunction,Coverlay, {Top|Bot}	
FileFunction,Heathsink,{Top|Bot}	
.FileFunction,Paste,{Top|Bot}	

The drill and rout layers

.FileFunction,NPTH	Non-plated through-hole drill file
.FileFunction,PTH	Plated through-hole drill file
.FileFunction,Drill,i,j	Blind or buried via drill file from I to j; always plated
.FileFunction,NPRoute	Non-plated route
.FileFunction,PRoute	Plated route
.FileFunction,Scoring	
.FileFunction,Backdrill,i,j	

It is general practice to put blind and buried drills in different files. We followed that practice. For NPTH and PTH there are two systems in use: separate files, or a single file with NPTH and PTH tools. We selected to encode the NPTH/PTH information by split files for consistency with the way blind and buried and with the concept that where image is present is defined by a file and its file attribute.

The profile layer

This layer defines the outline or profile by drawing it with a zero size attribute. The outside edge of the board must be defined in this file. Openings in the board can be defined in this file or in one of the route files. Plated edges are rare but exist, and they can be expressed by the{P|NP}.

.FileFunction, Profile, {P|NP}

The documentation layers

These layers do not represent a physical part of the printed circuit board. These are drawings to help in manufacturing and provide extra information.

.FileFunction,Drillmap	
.FileFunction,Assembly	Assembly drawing
.FileFunction,Glue, {Top|Bot}	
.FileFunction,Mechanical	A mechanical drawing
.FileFunction,Pads,{Top|Bot}	
.FileFunction,KeepOut	
.FileFunction,Drawing, <free text>	Any other manufacturing drawing

The other layer

The “Other” caters for functions that were overlooked. It cannot be used for any of the functions .explicitly included. The free text can be used to describe the function of that layer. The list of function attributes will be extended if the need arises.

.FileFunction,Other, <free text>	None of the above

On the file name

The function is associated with the whole file. Any file attribute can also be expressed in the file name rather than with an attribute. One does not exclude the other.

The file function attributes can be used to define a file naming specification. The name consists of the following elements, separated by a “_”. Files have the extension “.gbr”; the extension then indicates the format used as is the general convention.

a free prefix to define job (name, revision, origin, date etc.
the context attribute
the function attributes

An example: Phifac$Rev27b_Array_Soldermask_Top.gbr

Note that attributes associated with apertures or objects cannot be expressed in a file name. Attributes are anyhow needed.

On materials

The attributes defined here do not cover the materials; therefore describe only the layer structure and not the full stackup. This is intentional. The materials are not associated with an image, and

therefore do not naturally fit with Gerber files. Adding materials is not a straightforward extension of the Gerber format. Our current position is that materials are better transferred by a subset of IPC-2581 standard. This is the only standard that exists even if it is not used very often. There is no need for a second new standard. In such an IPC-2581 context the file name attribute is akin to the *layerFunction* in the *Layer* element.

On components

The attributes defined here do not cover the materials. This is intentional. Components do not naturally fit with Gerber files, except maybe for the footprint. Our position is that materials are better transferred by a subset of IPC-2581 standard.

On the netlist

The attributes defined here do not allow to transfer a netlist as a Gerber file. This is intentional. Certainly, attributes to specify a netlist can be defined. However, theIPC-356-A standard is perfectly suited to transfer netlist information. Most systems have an IPC-356-A input and output, albeit of uneven quality. Our current position is that introducing a new format for netlists is not needed and would be a distraction. What is needed is that all archives include an IPC-356-A netlist and that the quality of some implementations is improved. (Note that introducing a new format does not solve implementation problems. In the contrary, the newer implementation will add teething problems of its own. Implementation problems are solved when users complain and software suppliers act on these complaints.)

The File Subject Attribute

The **.FileSubject** file attribute identifies the subject of the file. It can take the following values:

.FileSubject,SinglePCB	Single PCB
. FileSubject,Array	Array, Customer Panel, Shipment Panel
. FileSubject,ProductionPanel	Production Panel, Shipment Panel, Fabrication Panel
. FileSubject,Coupon	Test Coupon
. FileSubject,Other, <free text>	None of the above

The subject attribute is to step and/or context in IPC-2581.

The File Version Attribute

The **.FileVersion** file attribute identifies the Gerber format version used in the file.

.FileVersion,<Decimal Number>	The decimal number is the version number.

From this extension the attribute is mandatory when attributes are used, and it must be the first attribute. The version number of this extension is “2.1”. Its presence warns the input processor that the file is of a higher revision. The current version of the Gerber format is deemed to be “1.1”; the attribute did not exist and therefore is not present in these files.

The File Comment Attribute

The **.FileComment** file attribute allows to further clarify the purpose and use of the file.

.FileComment,<Free text>	The decimal number is the version number.

The Gerber format already has the G04 comment code. However, this is often used to add comments to constructs in the file. The .FileVersion attribute serves to store information about the application and use of the whole file. A Gerber file reader may for instance display this file comment. There is a similar concept in IPC-2581.

The Aperture Function Attribute

The **.AperFunction** aperture attributes defines the function of all graphics objects created using this attribute. Consequently, all objects of the same aperture have the same function. A different function requires a different aperture, even if the apertures coincidentally share the same shape.

Note that graphics objects can also be created without apertures, namely regions. The equivalent **.RegiFunction** object attribute defines the function of all regions. For clarity we will focus on the aperture based object only for the moment.

For each file function there are only a limited number of functions that make sense. E.g. assigning the function of via to a drill hole makes sense, assigning SMD pad does not.

The Drill and Rout File Function Attribute Values

The **.AperFunction** aperture attribute defines the function of drill holes created with that tool. It can take the following values:

.AperFunction,ComponentDrill	Holes used for component pins, can also serve as via
.AperFunction,ViaDrill	Holes with only the via function
.AperFunction,MechanialDrilll	Holes used for mechanical reasons, e.g. screws
.AperFunction,OtherDrill,<free text>	Holes used for none of the above

These values can only be set on apertures representing a drill tool. Its value is applied to all holes drilled with that tool. All holes drilled with the same tool have the same function – mixed tools are intentionally not supported.

The function “Other” caters for functions that were overlooked in this specification. The set of functions will be extended if the need arises. Note that “Other” cannot be used for any of the functions explicitly included; e.g. “Other” cannot be assigned to a via tool.

Note: There is no method to represent mixed tools. This is intentional as they are problematic in CAM. It may be objected that e.g. a component and via tool have the same diameter, and that it is therefore more efficient to use the same tool; not true; use two tools, having the same diameter, and let CAM do any desired optimizations for drilling; the purpose of this tool is not to optimize

drilling but to streamline the transfer of a design from CAD to CAM. Another disadvantage of associating this attribute to individual drill holes is that it would blow up the file size.

The Drill Tool Parameter Attribute

The **.DrillToolParameter** aperture attributes is used to define size parameters associated with a drill tool. Sizes are decimal numbers, expressed in the Geber file unit defined by %MO. Note that the aperture diameter expresses the end diameter of the hole, and not the tool diameter.

.DrillToolParameter,TolerancePlus, value	
.DrillToolParameter,ToleranceMinus, value	
.DrillToolParameter,PlatingThickness, value	
.DrillToolParameter,Filled, {yes|no}	
.DrillToolParameter,Other, <free text>	None of the above

The Top and Bottom Copper Layer Flash Function Values

The **.AperFunction** aperture attribute specifies the function of flashes created with that aperture. Its purpose is mainly to define the function of pads (aka lands). It can take the following values:

.AperFunction,Component	
.AperFunction,SMDPad,{SMDef|nSMDef}	SMDef means Solder Mask Defined
.AperFunction,BGAPad,{SMDef|nSMDef}	
.AperFunction,TestPad	
.AperFunction,ConnectorPad	
.AperFunction,ViaPad	
.AperFunction,FiducialPad	
.AperFunction,OtherPad	
.AperFunction,Pattern	For patterns, e.g. to balance copper
.AperFunction,Other, <free text>	Any other flash function

The “Other” caters for functions not foreseen in this list. It cannot be used for any function in the list.

Note that to apply this attribute all pads must be flashed, or, if painted, identified by the .PaintedFlash attribute (see below).

Painted Flashes

CAM must know where pads (lands) are and what their function is. (Some examples. One need pad locations to generate the netlist. Whether a pad is a fiducial or an SMD pad affects the solder mask.)

The natural way to transfer pad functions from CAD to CAM is by an attribute assigned to a flash. To do this, there must be a flash. The natural way to identify pads in the file is by using a single flash for each pad, so called ‘flashed pads’. However, some Gerber files unfortunately do not use flashed pads but ’painted pads’, i.e. pads constructed by a number of draws. Painted pads do produce the correct image, but there is no single object to identity the pad. Painted pads are intensely hated by

CAM engineers because the information of what is a pad is lost and must be manually recreated, a time consuming and error prone process.

Note that the use of painted pads is not due to any weakness of the Gerber format. Painted pads occur in all formats. In fact, the powerful macro aperture statement is a unique feature of the Gerber format and allows creating apertures of any shape easily. One is not limited to pre-defined aperture shapes.

As stated before, the best way to identify pads is by using flashes. However, we want to make it easy for existing implementations relying on painted pads to add flash function values to the file, without compelling them to first replace all painted pads by flashed ones. For this purpose we introduce the .PaintedFlash *object attribute*, with the following syntax:

.PaintedFlash, <apertureID>, <flashID>

The integer <apertureID> identifies the virtual aperture (shape) that is flashed. The integer <flashID> identifies an individual virtual flash object; all objects constituting the same pad must have same <flashID> (and of course the same <apertureID>.

With the .PaintedFlash object attribute the objects are uniquely identified and function attributes can be associated with them.

Note that the .PaintedFlash attribute should be viewed as a transitional attribute only. Eventually, all pads must be flashed.

Discussion

The attributes discussed here transfer essential information needed to manufacture a PCB in a standard and machine readable form.

Note that some information is not handled by these attributes.

The new parameters do not describe the stackup of materials. As materials are not image related, we feel there is not much need to transfer this in a Gerber style format. The stackup subset of the new IPC-2581 format exists, is well defined and can be easily tweaked to do this job in a Gerber context.

The new parameters do not describe impedance controlled tracks. This is image related and could be a future extension or dealt with via IPC-2581.

The new parameters do not describe the netlist. We feel there is no need. The IPC-356-A standard is adequate. An IPC-356-A netlist can be combined with the Gerber files in the same archive. CAM systems can process IPC-356-A. We see no added value in inventing a Gerber-like netlist format, or any other new netlist format for that matter. If it ain't broke, don't fix it.

None of the new parameters affect the image. This is a major benefit. A Gerber file extended with attributes is essentially compatible with the existing Gerber readers: the readers may throw an error such as “Unknown Parameter” but the correct image is generated. The error message actually shows useful information, such as the file function. *This extension of the Gerber File Format is compatible with existing workflows.*

The most complex part of the description of a PCB is the image. An error in the image is hard to detect, and usually results in scrap. Geometric software is hard to get right and takes a long time to debug, often years. This extension does not change the geometric part of current Gerber based data transfer, both for writers and readers. One can continue to use the trusted and thoroughly field tested images.

With this extension the Gerber file maintains its key benefit of being simple and human readable.

Implementing the new parameters is straigntforward. Gerber writers know which file they are writing, adding a line such as “%TA.FileFunction,Copper,Top,1*%“ is straightforward. For a CAM system it is as simple to parse that line and use the information to put the image in the correct place in the layer structure. The chances for bugs are low, and if it might happen the CAM operator will probably notice. The chances of scrap are negligible. In fact, the chances of error are lower than if a CAM operator must manually assign a function to the layer. The same holds largely for aperture function and drill tool parameters.

Adopting the new attributes has a very low risk and implementation cost, but has major benefits.

Acknowledgment

This draft was reviewed by a group consisting of Luc Samyn, Wim De Greve, Roland Polliger, Arnold Wiemer, Ludek Brukner, Dirk Leroy, Rik Breemeersch.

Call for Comments and Additional Attribute Proposals

In this proposal we have defined a first set of standard attributes with basic characteristics of a PCB. This proposal does not attempt to be complete and define all possible characteristics of a PCB.

We call on the PCB design and manufacturing community to comment on this proposal, and, more importantly, propose other attributes handling characteristics not covered here. All comments will be considered.

Comment can be sent to gerber@ucamco.com

If there is support for this proposal we will publish an extension to the Gerber Format based on it.

Copyright, Intellectual Property and Trade Name

Ucamco developed the Gerber Format and improves it from time to time with updates. The Gerber Format is Ucamco intellectual property. No derivative versions, modifications or extensions can be made without prior written approval by Ucamco. Developers of Gerber software must make all reasonable efforts to comply with the latest specification.

Gerber Format is a Ucamco trade name. Users of Gerber Format will not rename it, associate it with data that does not conform to the format or modify the graphical interpretation of the format.

Correspondence regarding this publication can be sent to:

gerber@ucamco.com

or

Ucamco NV
Bijenstraat 19,
B-9051 Gent,
Belgium

For more information see www.ucamco.com